Experimenting with Image Priors in Super-resolution CNNs

Matthew A. McCready
Department of Electrical Engineering, Stanford University
450 Serra Mall, Stanford, CA
mattmc@stanford.edu

Abstract

Single-image-super-resolution seeks to transform a single low resolution input image into a corresponding high-resolution output image. This problem lends itself well to convolutional image-to-image deep-learning approaches. Few such approaches have explicitly trained their networks on natural image priors. Here, an investigation of CNNs for super-resolution is presented with anisotropic total variation of network outputs included directly in the training loss. It is shown that such networks are capable of sharpening low-resolution inputs for both sysematically and generally downsampled images.

1. Introduction

Super-resolution can be used to transform a low-resolution image to a high-resolution image. This ability has many uses in everyday life, healthcare, and tech. For the general public this can include improving resolution of childhood photos from when photographic hardware was less sophisticated, or even taking high digital zoom photos from smartphones. In the science and tech communities this can mean higher compression of images can be stored and transferred with recoverable HR originals. Super-resolution can even contribute to saving lives by allowing for rapid low resolution MRI images to be taken and true HR details to be recovered.

Super-resolution techniques fall into two general categories: multi-image super-resolution and single-image super resolution (SISR). In this work the focus will be on SISR, as the former requires multiple input LR images of a single scene and is therefore far less convenient than SISR.

While SISR can be carried out with classical reconstruction and interpolation methods, the problem lends itself well to deep learning approaches. In this work we will apply a convolutional neural network (CNN) to low resolution input images, to produce corresponding high-resolution output images. We experiment with the inclusion of image priors in the loss function similar to reconstruction methods, and design networks for SISR using full RGB images and using the luminance-channel only. We will see that the use of such networks can sharpen low-resolution images and improve peak-signal-to-noise ratio by up to approximately 3 dB.

2. Related Work

2.1. Single Image Super Resolution

SISR seeks to solve the ill-posed problem of recovering some high-resolution image HR given a single low-resolution image LR. It is generally the case that LR is related to HR through blurring, down-sampling, and addition of some noise. Classic attempts at solving this problem rely on interpolation or reconstruction models. Interpolation models such as bicubic interpolation (BI) can provide quick results which greatly improve over a naïve zero-order hold [5]. Reconstruction methods rely on some image prior in order to solve the underspecified ill-posed problem [10]. These solutions are often time-consuming iterative processes which react poorly to high down-sampling ratios.

2.2. Deep Learning for Super Resolution

The current state of the art approaches to SISR are provided by deep-learning models. These methods fall under "example based" methods which learn from a set of LR and HR pairs. A staple and benchmark of SISR DL models has been super-resolution CNN (SRCNN) [2, 3]. This network pre-interpolates the LR image using BI, then feeds the result through 3 convolutional layers, maintaining image dimensions. This has the benefit of use for any scale of up-sampling as the up-sampling is performed before input to the network. Networks which perform up-sampling themselves using deconvolution have been implemented such as FSRCNN [4]. State-of-the-art SISR uses Very deep models such as VDSR and EDSR which have shown improvement without use of batch normalization [6, 9]. The architecture for such networks often relies on skip connections to prevent vanishing gradients.

Introducing attention can also provide benefit, as some

regions of the image are of higher importance in the SISR problem than others. For example, a plain background needn't change much from LR to HR, but higher spatial frequency regions such as edges do. SelNet uses "selection units" to gate what information from a convolutional block should be passed on [1]. Generative models such as SRGAN have also become popular [7]. These can use similar structures and loss functions to other models with the added benefit of adversarial loss from a discriminator.

2.3. Image Priors

As mentioned in section 2.1, reconstruction methods rely on known image priors to converge on a solution. For deep learning models, engineers can attempt to incorporate image priors into their network architecture or learning algorithms to improve performance. For example, a SRCNN architecture has been developed with a feature extraction layer paying close attention to image gradients and included in an additional term of the loss function [8]. In this work we will investigate this prior incorporation improvement by minimizing the image gradients of the output image directly in the loss function.

3. Methods

In this project the implementation of an SRCNN based architecture will be investigated. This will be performed with multiple loss functions including the vanilla mean-squared-error (MSE) loss function, and MSE with an image prior loss function to gauge any benefit.

3.1. Baseline Method and Metrics

The baseline method for comparison is simple bicubic interpolation. BI works by performing a degree-3 bidirectional fit to the image for interpolation. Each local fit is made over a 4x4 region of pixels. BI is applied to all LR image patches in the dataset for comparison with SRCNN approaches. In addition to providing the baseline method, BI will be used to scale up LR images for input to the SRCNN models. This is to take advantage of the SRCNN architecture which maintains the input image dimensions at output. With this method the SRCNN can be applied to arbitrary scale up without the need for specialized network architectures.

Some of the most common metrics for evaluating SISR performance come from standard image comparison metrics; peak-signal-to-noise-ratio (PSNR) and structural similarity index measure (SSIM). These will be used for evaluation of predicted images. PSNR is given in (Equation 1) where MSE is the mean-squared-error discussed in section 3.3.

$$PSNR = 20 \cdot \log_{10}(I_{\max}) - 10 \cdot \log_{10}(MSE) \quad (1)$$

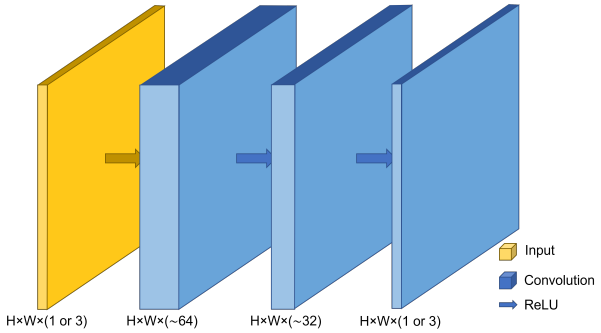

Figure 1. SRCNN Architecture. Input can be of arbitrary height (H) and width (W) which are maintained throughout the forward pass. Input and output will have 1 or 3 channels depending on application to RGB channels or luminance only. Yellow layer and arrow represent input, blue layers represent results of convolution, and blue arrows represent ReLU non-linear activation functions. Layers 1 and 2 are chosen near 64 and 32 channels respectively.

In this work predictions will be of type float32 scaled and clipped to the range [0, 1] before PSNR calculation. This results in an $I_{\max}$ of 1.0 reducing the first term in the above equation to zero. SSIM perceives image degradation as change in structural information of the image. This is calculated by applying (Equation 2) to various windows across the image.

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (2)$$

Here, x and y are the windows of the predicted and ground truth images respectively, μ is the average of the window, σ^2 is the variance of the window, σ_{xy} the windows' covariance, and c_1 and c_2 stabilization variables. These metrics are calculated and averaged on a large batch size efficiently using the PyTorch-Ignite package [11]. SSIM uses default settings.

3.2. SRCNN Architecture

The SRCNNs implemented here will begin with the architecture of the original, using 3 convolutional layers with ReLU activation after layers 1 and 2 for non-linearity (Figure 1). The model is initialized with the parameters of the larger receptive field SRCNN from Dong et al [3] using f1 = 9, ch1 = 64, f2 = 5, ch2 = 32, f3 = 5 where f and ch refer to filter size and number of channels and the suffix number refers to the corresponding layer. Networks were implemented using the PyTorch Module API [11].

These values were varied stochastically in local ranges and the resulting models trained for 25 epochs. The best model is retained for further training and evaluation. The best model calculated on MSE loss will be included in the architecture search of the image prior loss model. Batch-norm layers were not included as it has been shown that net-

works for SISR perform better without them [6, 9]. Regularization will be provided through weight-decay discussed in section 3.4.

3.3. SRCNN Loss and Image Priors

The baseline implementation of SRCNN will use MSE as a loss function given by equation (2).

$$\text{MSE} = \frac{1}{2N} \sum_{i=1}^{N} \sum_{j,k,c} \left(\hat{y}_{jkc}^{(i)} - y_{jkc}^{(i)} \right)^2 \tag{3}$$

Where N is the number of training examples multiplied by the number of elements in an image, $\hat{y}$ is the model output, and y is the ground truth HR image. The gradient of this loss function for training example i is given by:

$$\nabla_{\hat{y}} \text{MSE}^{(i)} = \frac{1}{N} \left(\hat{y}^{(i)} - y^{(i)} \right) \tag{4}$$

In addition to MSE loss, this project will experiment with the inclusion of image priors in the loss function. In this course we have discussed regularization of network weights but not regularization of the network output. This makes sense as we have typically dealt with classification problems. Here, however we are addressing a reconstruction problem where the output is an image. We now ask if we can use prior knowledge of natural image features to incentivize the network to produce natural looking images in less training time.

For this purpose, an additional term is included in the loss function, giving the anisotropic total variation (TV) of the output image (Equation 5).

$$\text{TV} = \frac{\lambda}{N} \sum_{i,j,k,c} \left(D_x \hat{y}^{(i)} \right)_{jkc}^2 + \left(D_y \hat{y}^{(i)} \right)_{jkc}^2 \tag{5}$$

Where the operators D_x and D_y represent the horizontal and vertical gradients applied to the output image respectively, and λ is the TV norm term weighting. Natural images tend to have sparse gradients. Teaching the network to minimize total variation should therefore improve performance. The gradient of the TV norm term for example i is given by:

$$\begin{aligned} \nabla_{\hat{y}} \text{TV}^{(i)} &= \frac{2\lambda}{N} D_x^T \left(D_x \hat{y}^{(i)} \right) + D_y^T \left(D_y \hat{y}^{(i)} \right) \\ &= \frac{4\lambda}{N} \left(D_y D_x \hat{y}^{(i)} \right) \end{aligned} \tag{6}$$

Where the expression has been simplified using the fact that the horizontal finite difference operator becomes the vertical operator when transposed and vice versa, as well as Clairaut's theorem which says that derivatives are commutative.

3.4. Optimization and Hyperparameters

For this project the stochastic gradient descent Adam optimizer update scheme is used for training the networks on minibatches. The Adam update requires four hyper parameters: the exponential decays for the first and second moments β_1, β_2 which were set at their default values of 0.9 and 0.999 respectively, the numerical stability term ϵ typically set to a very small value such as 10^{-8}, and the learning rate initialized here at 10^{-3}. Through trial and error it was found that optimization performed better with an initial learning rate of 10^{-4}.

Although Dong et al [3] note that the SRCNN achieves greater performance with the final layer's learning rate set to one tenth that of the others, this detail is unnecessary when using Adam as opposed to their SGD + Momentum optimizer since Adam should reach the appropriate learning rate on its own. Indeed, through preliminary tests inclusion of the parameter specific learning rate yielded no improvement. A learning rate schedule was included however, in an attempt to force the learning rate to decrease after evaluation metrics plateau during training. The schedule reduces the learning rate by a factor of 10 after the 30th and 80th epochs.

The weights and biases of the 3 convolutional layers were initialized with Kaiming-normal initialization. This uses a normal distribution with mean zero and standard deviation $\sqrt{2/D_{in}}$ where D_{in} is the total dimension of the weight given by the product of the squared filter size and the number of incoming channels. This is chosen over Xavier initialization to maintain activations as this model utilizes ReLU nonlinearities. During training, L2 regularization is applied to the weights and biases in the Adam optimizer with a weight of 10^{-4}. This promotes generalizability, removing risk of overfitting and exploding gradients. The L2 regularization term is not included in the explicit loss calculation for the results shown in section 5.

The minibatch size was chosen to be 25 after experimentation with smaller values. Going any higher resulted in occasionally errors being thrown due to lack of GPU memory depending on the number of filters used. Filter sizes and amounts were varied as described in section 3.1. The weighting on the TV loss term was initialized as 10^{-3} to start and varied within a local set of values for a small number of epochs before settling on 10^{-4} for better results.

4. Dataset

In this project all methods are performed using the Flikr2K dataset constructed for the NTIRE 2017 SISR challenge [9, 12]. The dataset contains 2650 HR images of 2K resolution. These images are very large (4.4MB on average) with many details and would be computationally expensive to learn from directly. In order to lower training

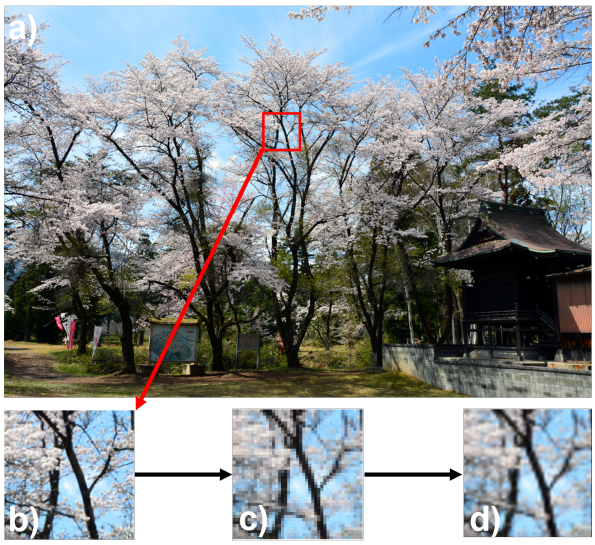

Figure 2. Training dataset and baseline generation. a) The full HR 2K image. b) The HR image patch corresponding to the red box of the original 2K image. c) The LR image blurred and down-sampled according to (Equation 7). d) The baseline up-sampled BI image used for input and comparison.

time and generate more training pairs, the images were split into patches of size 128 × 128 (3.4KB on average). The process can be seen in (Figure 2).

Taking 10 patches from each example yields 26,500 HR images. These are then split into 24,000 training, 1,000 validation, and 1,650 image patch test sets. The corresponding LR images are generated by applying a Gaussian blur kernel K of standard deviation σ_{blur} = 1, down-sampling by a factor of 2, 3, or 4, and applying Gaussian additive noise $\mathcal{N}$ of mean μ = 0 and σ_{add} = 1 (Equation 7).

$$LR = (HR * K(\sigma_{blur})) \downarrow + \mathcal{N}(\mu, \sigma_{add}) \qquad (7)$$

The problem of SISR can be difficult to apply directly to a 3 channel RGB image. This formulation would require mapping from a H × W × 3 image to another H × W × 3 image. This implementation has been carried out [3], however, it is common to reduce the SISR problem to a simplified single channel yielding similar final results [2].

This can be done by converting the given LR and HR RGB pair into YCbCr space and applying the SRCNN to the luminance (Y) channel only, with the chrominance (CbCr) channels upscaled using bicubic interpolation only. This approach has been shown to be effective as resolution information is mainly stored in the luminance channel, and applying SRCNN to the chrominance channels gives little improvement [2, 3].

The inclusion of Gaussian blurring and noise could also further complicate the problem, as the network now needs to perform deconvolution and denoising. Many previous implementations of SRCNNs simply downsampled the HR data directly to generate LR data. Furthermore, they trained individual networks (albeit with the same architecture) for different factors of upscaling [2–4].

To combat these increased difficulty factors, the training, validation, and testing data sets were split into two groups. One applied Gaussian blurring and noise as described above and consisted of a total of 26,500 images in RGB space with downsampling factors of 2, 3, and 4. The other consisted of 2,650 purely factor 3 downsampled images with no blurring or additive noise, and was transformed to YCbCr space (post-BI) for SRCNN application to the luminance channel only.

5. Results and Discussion

5.1. Baseline Bicubic Interpolation Results

The Flikr2K dataset was patched into 26,500 RGB images and split into test, validation, and training datasets as outlined in section 4. The HR image patches were transformed to LR patches via (Equation 7) to create the necessary training data.

The baseline BI was applied to all LR image patches to establish performance for later comparison. This also generates the inputs for the SRCNN architecture. The resulting average PSNR and SSIM metrics for this baseline are reported in (Table 1).

Set	PSNR	SSIM
RGB	24.35	0.73
Luminance	24.70	0.71

Table 1. Average PSNR and SSIM values from BI baseline on test sets of full RGB and luminance-only image patches

The Flikr2K set was also patched into a collection of 2,650 images and split into test, validation, and training sets. These HR patches were purely downsampled with a factor of 3 without any blurring or additive noise. The HR and LR patches were converted to YCbCr space and BI was applied as before. The resulting average PSNR and SSIM metrics for this baseline were calculated and reported in (Table 1).

The results of the BI baseline applied to these sets of LR patches are used as input data for the SRCNNs in the following sections.

5.2. SRCNN Hyperparameter Searches

Networks were trained with locally varied filter parameters as described in section 3.2 for a starting period of 25 epochs on the luminance-only data set for MSE-only and MSE+TV losses respectively. The results of the MSE+TV parameter search are plotted in (Figure 3) giving the losses,

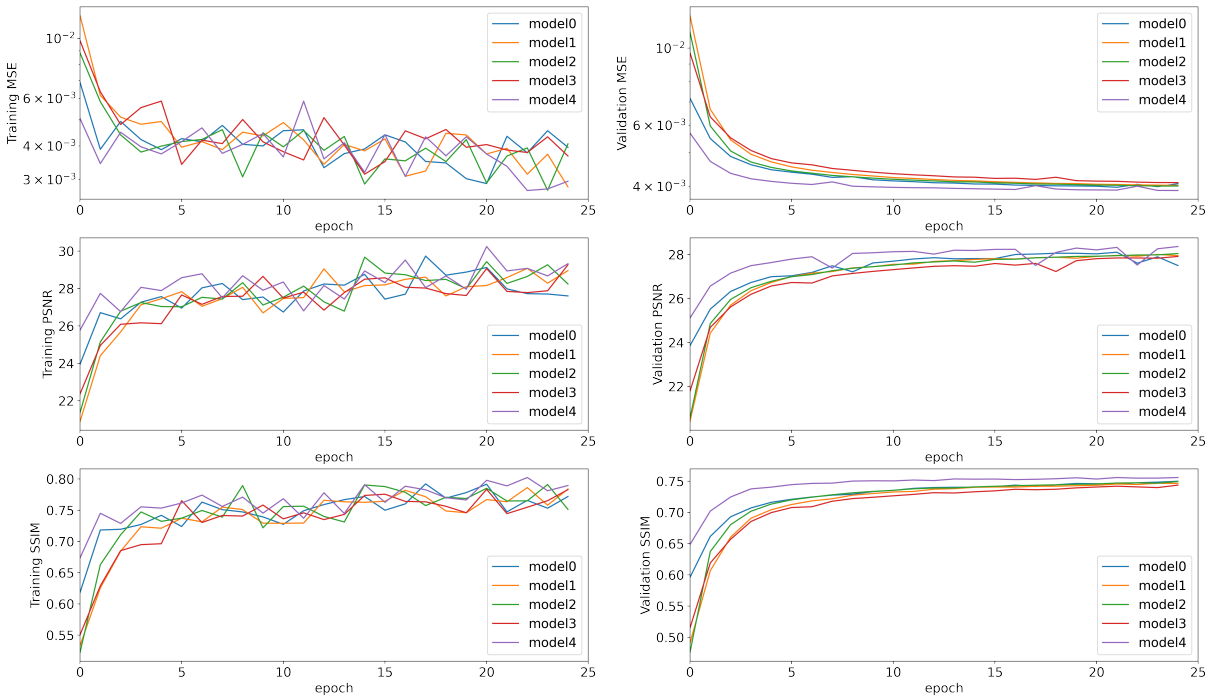

Figure 3. Hyperparameter search for MSE+TV Trained models on luminance-only data. 5 models were trained for 25 epochs with varied filter hyperparameters as described in section 3.4. Model 0 gives the initial baseline model from Dong et al [3]. MSE loss (TV not included), PSNR, and SSIM values on training and validation data are plotted over the training epochs.

average PSNRs, and average SSIMs for the training and validation sets.

As can be seen from the validation results there is a clear correlation between MSE loss and the PSNR metric. This is to be expected given the maximum image value of 1.0 setting the first term of (Equation 1) to zero. SSIM performs similarly but with less clear dependence. It can be seen that the model performs better on the training set than the validation for all metrics, as is to be expected. However, the difference is very slight, indicating that no overfitting to the training set is occuring.

Curiously, the training metrics display much greater noise across epochs than the validation counterparts. This is likely due to how the training metrics were calculated. Due to limited memory, training metrics were not calculated across the entire training data set, but rather a subsection of equal size to the validation set. As the training data was shuffled prior to each new epoch, the images in the subsection change prior to each calculation. The validation set does not encounter this issue as it is small enough to perform the calculations on the full set at each epoch.

The resulting best models for the MSE and MSE+TV parameter searches were retained and trained for an additional 100 epochs with learning rate scheduled decay after epochs 30 and 80 at a factor of 0.1. These models had filter parameters decided be the search to be f1 = 8, ch1 = 61, f2 = 3, ch2 = 20, f3 = 3 for the model based purely on MSE, and f1 = 7, ch1 = 53, f2 = 6, ch2 = 25, f3 = 5 for the model based on MSE+TV. It was found that using a filter size of 1 (equivalent to a fully connected layer) gave the worst results, in particular when used in the last convolutional layer.

Little additional performance was gained through further training and the learning rate scheduling did not appear to have much effect, likely due to the use of Adam for adaptive learning rate. The MSE and MSE+TV models are compared for losses and PSNR on training and validation sets in (Figure 4). Again we can see that while training performance is slightly better than validation the models are not overfitting. We can see that the MSE+TV loss trained model appears to learn slightly faster than the MSE-only model, though they reach nearly the same final performance.

Training on the full RGB dataset produced largely the same behavior with the best model on the MSE only dataset being formed from the parameters f1 = 7, ch1 = 55, f2 = 6, ch2 = 20, f3 = 3, and the best model on the MSE+TV dataset from the parameters f1 = 9, ch1 = 56, f2 = 3, ch2 = 23, f3 = 6. These models were trained for fewer epochs (25 in total) due to the large amount of training data.

5.3. Resulting Reconstructions and Metrics

After a fully trained model was acquired it was applied to the corresponding test set of genaeralized downsam-

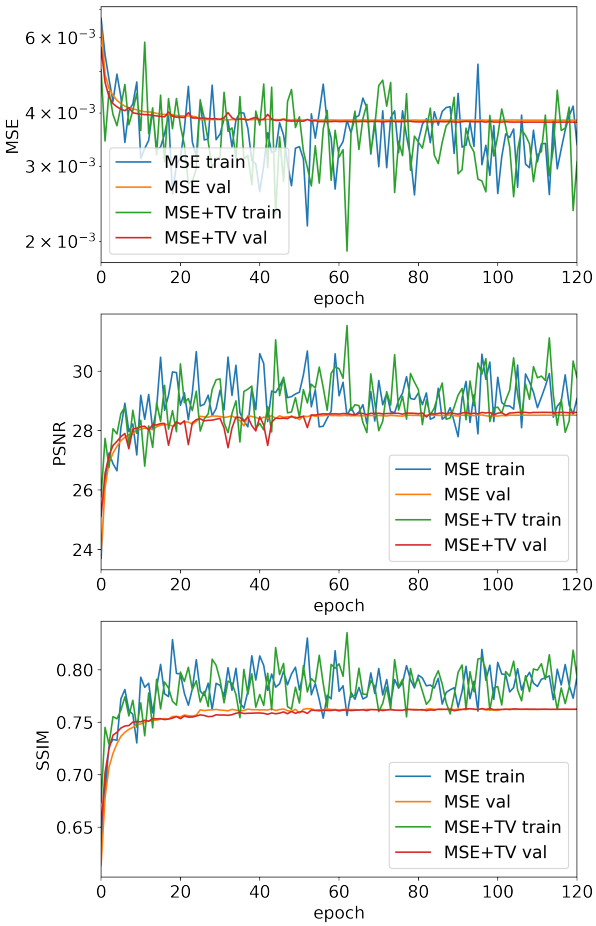

Figure 4. Training comparison for MSE and MSE+TV models on luminance-only data. MSE (not TV) loss, PSNR, and SSIM are plotted over 120 training epochs for both models on training and validation data.

pling RGB or simplified downsampling luminance-only images. The average PSNR and SSIM metrics were calculated across these test sets and are reported in (Table 2).

Model	PSNR	SSIM
Luminance MSE	28.59	0.78
Luminance MSE+TV	28.49	0.78
RGB MSE	27.47	0.75
RGB MSE+TV	27.95	0.76

Table 2. Average PSNR and SSIM values from best SRCNN models on test sets of full RGB and luminance-only image patches.

The final PSNR and SSIM values show that integration of the anisotropic TV norm with MSE loss does not converge to a better solution than MSE alone, though it does give nearly the same results.

If we visually compare the results of SRCNN models

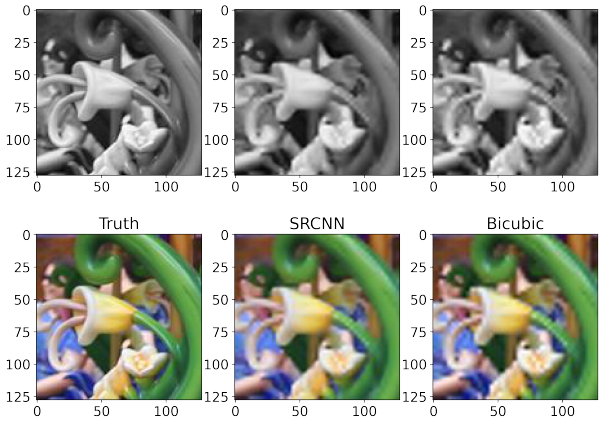

Figure 5. Example test image result for luminance-only dataset. Top grayscale row gives luminance channel results, while bottom row gives luminance channel-results recombined with chrominance channels and transfered back to RGB space. From left to right: Ground truth image, MSE+TV SRCNN image, and BI baseline image.

with the BI baseline results and the ground truth HR patches we can get a better sense of the models' abilities. Results using the luminance-only data set and MSE+TV model are given in (Figure 5). We can see that the SRCNN result exhibits less noise than the naive BI baseline giving a smoother feel to the objects as in the ground truth HR patch. The SRCNN result aslo appears to slightly lessen aliasing artifacts present in the BI baseline. We can see that these improvements remain when the chrominance channels are recombined with the luminance channel, reinforcing that chrominance channels do not need to be high resolution. The results using no TV are nearly identical with arguably slightly less defined edges to the objects. Results using the full RGB dataset are shown in (Figure 6) and we can see that the image is sharpened in comparison to the BI baseline, as seen in the details of the tool and the grass at the bottom of the image.

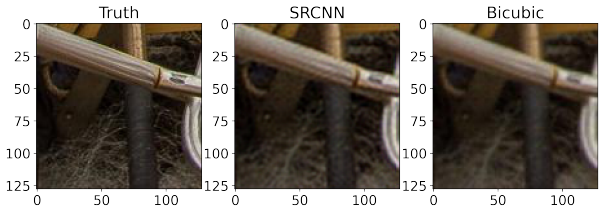

Figure 6. Example test image result for full RGB dataset. From left to right: Ground truth image, MSE+TV SRCNN image, and BI baseline image.

One very ill-posed case where the models seem to perform particularly poorly is when high degrees of aliasing are present. Aliasing occurs when data is sampled at a

rate below twice its highest frequency value (i.e. below the Nyquist rate). This means that images with high spatial frequency objects such as grating or other patters are greatly degraded during downsampling and are difficult to recover. An example is shown in (Figure 7) where grating in the original image is highly aliased in the BI baseline giving lower frequency patter almost perpendicular to the original. Additionally, the pattern of the wood grain is almost entirely lost. We can see that the SRCNN is unable to reconcile these errors and merely smooths the image like a denoiser.

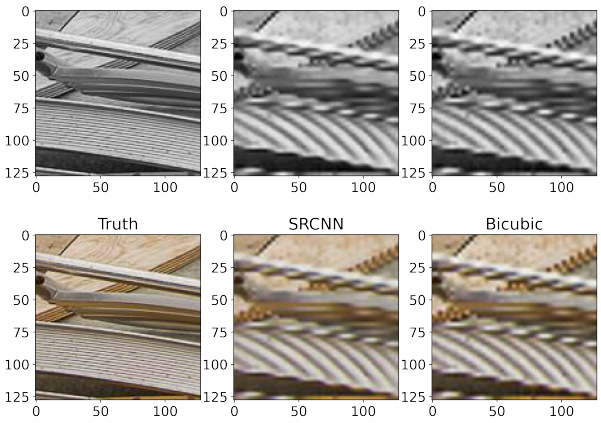

Figure 7. High aliasing example test image result for luminance-only dataset. Top grayscale row gives luminance channel results, while bottom row gives luminance channel-results recombined with chrominance channels and transfered back to RGB space. From left to right: Ground truth image, MSE+TV SRCNN image, and BI baseline image. Strong banding artifacts due to aliasing.

5.4. Limitations

This work has provided models with some improvement over the BI baseline but do not perform as well as other SRCNNs implemented in literature. This could in part be due to the choice of dataset, as the baseline SRCNN was implemented exactly as in literature with the only difference being the choice of data. The original SRCNN paper [3] trained on the ImageNet dataset which has an average resolution of 469x387 pixels often cropped to 256x256. This means that details in these images occupy less pixels than in the Flikr2K data set which contains images of resolution 2560x1440. This allowed the original authors to train on image patches of size 33x33, while this project used patches of 128x128 in order to maintain some structure on which to train. It is likely that this increases training difficulty. Furthermore, we can see that the BI baseline far under-performs on this dataset when compared to the datasets used by the original authors, so it is not surprising that the subsequent SRCNN results are also lower than the originals.

6. Conclusions and Future Work

SISR is a challenging problem which when solved can recover high resolution details from low resolution input images. As we saw in this work, an SRCNN can be used to improve results beyond that of a classical BI. This can be done through application to a luminance channel only, or a full RGB image. SRCNNs can better define edges in LR images and reduce small amounts of aliasing. It was shown that high aliasing can be very difficult to approach and such networks can offer little improvement on these ill-formed images. The inclusion of anisotropic TV image priors in training loss was investigated and shown to have little effect save for a slight potential at increasing training speed, though this work is hardly conclusive.

If this work were to continue, it may be better to utilize the datasets referenced in the original SRCNN papers, as the very high resolution of this dataset required larger image patches for training. Many networks for SISR have also been shown to produce better results with very deep architectures. This is much more computationally expensive but is in line with the current state-of-the-art. Many networks with the same architecture should be trained with and without TV included in the loss to further investigate if it provides any improvement. It would also be of interest to develop a data set of many images with high spatial frequencies, which exhibit extreme aliasing when downsampled. A network trained specifically for photographic anti-aliasing would be a very powerful computational imaging tool.

7. Contributions and Acknowledgements

The entirety of this work was completed by Matthew McCready with no collaborators. A personal computer was used for all computations with a NVIDIA GeForce GTX 1660 Ti GPU. The only outside code used were default MATLAB packages and Pytorch [11].

References

[1] J. S. Choi and M. Kim. A deep convolutional neural network with selection units for super-resolution. In *CVPRW*, pages 154–160, 2017. 2

[2] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 8692:184–199, 2014. 1, 4

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell*, 38(2):295–307, 2016. 1, 2, 3, 4, 5, 7

[4] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 9906:391–407, 2016. 1, 4

[5] R. G. Keys. Cubic convolution interpolation for digital image processing. 1

[6] J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016. 1, 3

[7] C. Ledig and et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 105–114, 2017. 2

[8] Y. Liang, J. Wang, S. Zhou, Y. Gong, and N. Zheng. Incorporating image priors with deep convolutional neural networks for image super-resolution. *Neurocomputing*, 194:340–347, 2016. 2

[9] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR*, pages 66–70, 2017. 1, 3

[10] A. Marquina and S. J. Osher. Image super-resolution by tv-regularization and bregman iteration. *Journal of Sci. Comput.*, 37(3):367–382, 2008. 1

[11] A. Paszke and et al. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2, 7

[12] R. Timofte and et al. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, pages 1110–1121, 2017. 3